The City College of New York

Grove School of Engineering

CSC 33600

Database Final project

Professor: Denis K.

Date: 5/22/2022

Abir Das

Riaz Ahmed

Derrick B. Duller

a) Motivation. Why did you pick this database?

We picked the student database because we are students ourselves. It will be easy to relate to and work with.

b) List the authors' roles: who was responsible for what part of the project.

We worked collaboratively to figure out our approach to making the database. To do this we joined a Zoom call so we can give each other feedback in real time. First we worked on a rough sketch of the ER Diagrams as it will help us plan and understand what we need to include in our database. Then we made the MySQL tables (projectcreatetables.sql). Then we divided up the work between each team member to make a function to perform tasks for each table. Riaz was in charge of the studentFunction(), Abir worked on the classesFunction() and Derrick handled the AAGFunction(). Our approach to each user performed tasks were similar as we all decided what to do collaboratively. Then we write the report.

c) Logic of the database you created: relations, their attributes, constraints, etc. Provide some toy examples.

The tables in the database is always designed to have a primary key. The table students has a primary key ID that auto increments as the table is populated. The table classes also has a primary key ID that auto increments as the table is populated. This makes sure that even if there are students or classes with different names, they will never be confused by the system. Lastly, attendanceandgrades table has 2 foreign keys being used as primary keys at the same time. The composite key consists of studentID that references the ID in students table and classID that references the ID in classes table. These two keys are made as the foreign keys to introduce a relationship between the attendance rate, the grades, the class, and the student. Since a student can only attend a specific class at a time, it makes sense to make them composite keys. This is the main relationship between the student and class. The relationship of attendance rate and grades to both of these variables is that a student has those data after they finish attending a class for that semester. The student is graded and also the number of times they have attended classes is analyzed and tabulated as data in the attendanceandgrades table.

d) List the modules that the program includes (headers, helper functions, etc.) and explain what functions they have. Explain what tables / relations your database is built on.

Part 1:

- import mysql.connector

- We are using import mysql.connector because we are connecting our python program to the database we created in MySQL Workbench. There we have the tables for students, classes, and attendenceAndgrades.

❖ import os

- We are using the import os module because we implemented a condition in our main function, which helps the user to quit the program after they are finished performing their task in the table.

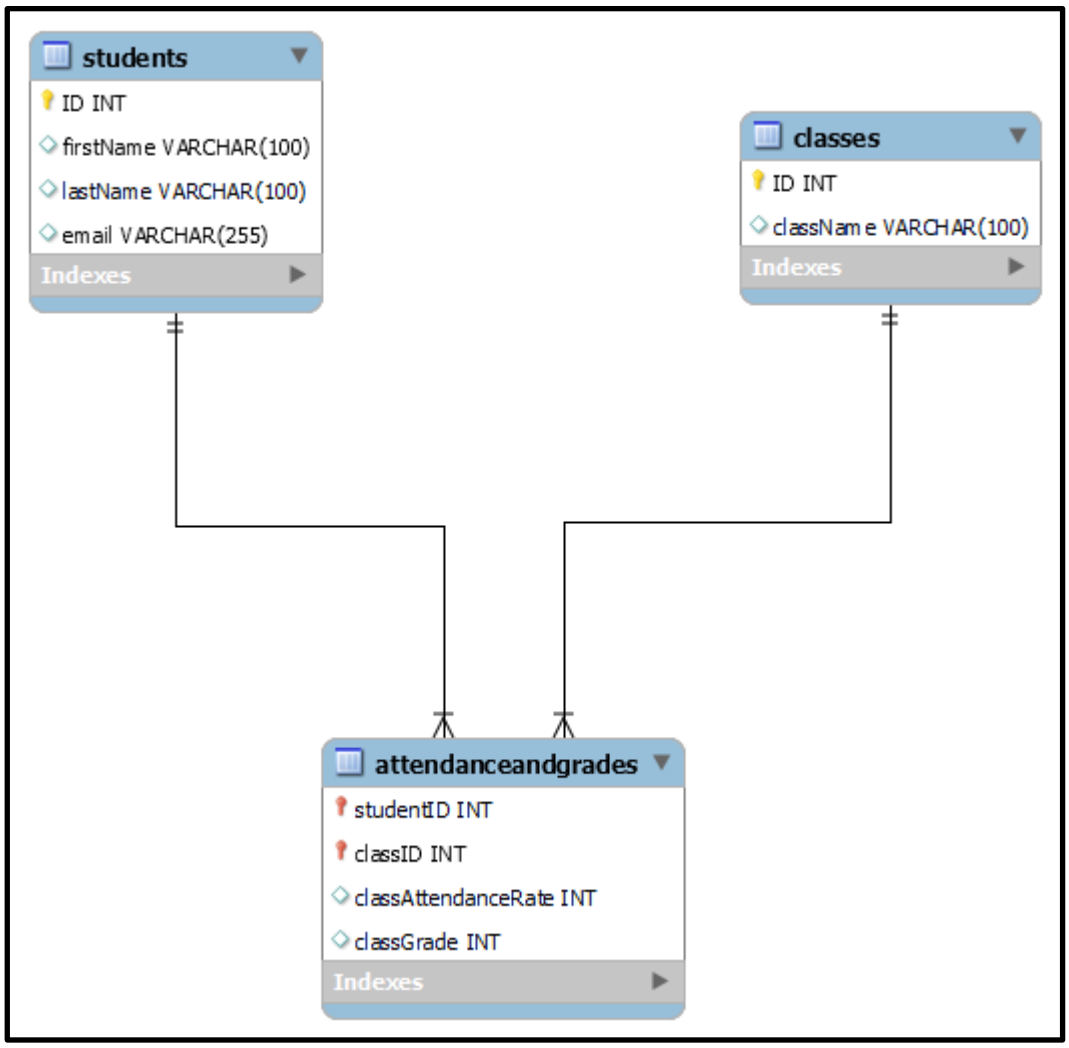
❖ import pandas as pd

- We are using the import pandas as pd module because in the assignment task we are required to implement a way to import a csv file by the user request for each tables. To do that we needed to use pandas. So now, when the user enters the location of their csv file in our program, their record gets added to the appropriate tables in the database on MySQL.

Part 2:

The tables we have in our database are the student table, classes table, and attendanceAndgrades table. In the student table, there is an ID column which is the primary key and it is auto-incremented. This means that the ID will be in the order of input by the user. Then there is a first name, last name, and email column for the students, which are strings inputs in the program. In the classes table, we have class ID, which is also auto-incremented and it is the primary key of that table as well. There is also a class name, which the user will input as a string. For example, the class name can be “Math 201.” Finally, we have the attendanceAndGrades table, which has int columns of student ID, class ID, class attendance rate, and class grades. In this table, the primary keys are student ID and class ID, and each is referenced as a foreign key to the student table and classes table. For example, student ID references the ID from the student table, and class ID references the ID from the class table. The attendance rate will be considered in percent form in the program.

e) ER diagrams.



f) Conclude with what you successfully managed to implement, and what difficulties you faced working on the project. What would you change in your code / algorithm to improve the program?

We successfully implemented a program where the user can interact with the 3 tables in our student database: the Students table (which holds the ID, firstName, lastName, and email of the student), the Classes table (which holds the ID and className of the class), and the AttendanceAndGrades table (which holds the studentID, classID, classAttendanceRate for that specific class of the student, and classGrade for that specific class of the student. The user can add new values, delete a value using the primary keys of the tables, display all of the contents of the table, fill up a simple query statement to select data on the table, and read a csv file to populate the table. These tasks are all operational for every table of the database. The difficulties lie with predicting how much an error can the user make with the inputs. If a phrase or word is wrong when

making the unique select query statement based on need, then the program fails. This is one example of a wrong input from the user. We solved it by use try and except in python programming. Basically, if there is an error, it will catch that error and still let us loop the function again so that the user can have another try instead of the program suddenly terminating. To further improve this program, we would like to deal with these errors a little bit better. Adding more options that limits getting specific input as lines for an sql statement to be executed by the code is the first thing that we would like to improve. However, that takes a lot of mastery and experience with python, so we did not do it for this project. However, we are indeed catching these errors to give the user another try on all the functions if something goes wrong with it. We also put a lot of warnings and basic formats that the user should keep in mind when using the program. Even with all these things lacking from the program, it is still fully functional and utilized.

***** The following screenshots show the outcome of our program. For this purpose, we are using the student table as a demo.**

Adding Values to the student table

```
 Welcome to the Student Database:
 1. student Table
 2. classes Table
 3. attendanceAndGrades Table
 4. Quit Program

Enter corresponding number task to be done: 1

 Students Table:
 1. Add Values
 2. Delete Values
 3. Display Table
 4. Make your own Select Query for students table
 5. Read CSV File to populate students table
 6. Back to Main Menu

Enter corresponding number task to be done: 1

Enter student first name: James

Enter student last name: Bond

Enter students email: JBond@yahoo.com

 Adding values done
```

In this picture, the user is in the main menu and they choose option 1, which is to perform a task on the student table. Then they choose what task they want to perform. Option one allows the user to add to student table. They add student “James Bond” with the email “JBond@yahoo.com”.

Displaying the student table

```
 Students Table:
 1. Add Values
 2. Delete Values
 3. Display Table
 4. Make your own Select Query for students table
 5. Read CSV File to populate students table
 6. Back to Main Menu

Enter corresponding number task to be done: 3
Contents of the table:

Student Table (ID (PK), FirstName, LastName, Email)
(6, 'Abir', 'Das', 'adas@ccny.cuny.edu')
(11, 'Henry', 'Riaz', 'HR@gmail.com')
(12, 'Derrick', 'Duller', 'dd@gmail.com')
(13, 'Johny', 'Mark', 'JM@gmail.com')
(14, 'Mark', 'Wood', 'MW@gmail.com')
(15, 'Jules', 'Mark', 'JM@gmail.com')
(16, 'James', 'Bond', 'JBond@yahoo.com')
```

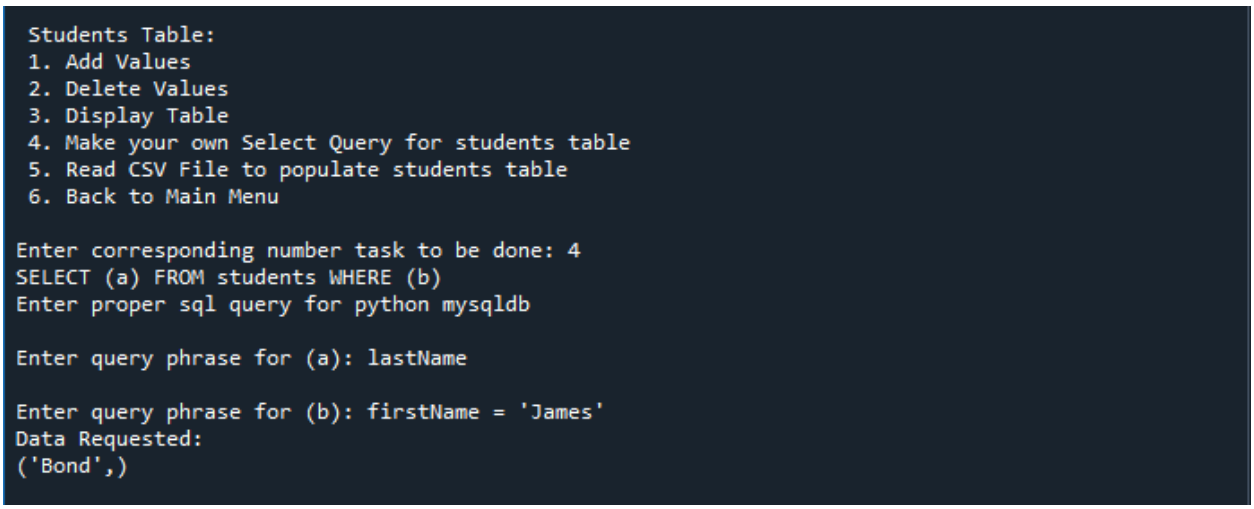
In this picture the user chooses option 3 to display the student table.

Delete ID from the student table

```
 Students Table:
 1. Add Values
 2. Delete Values
 3. Display Table
 4. Make your own Select Query for students table
 5. Read CSV File to populate students table
 6. Back to Main Menu

Enter corresponding number task to be done: 2
Student Table (ID (PK), FirstName, LastName, Email)

Enter Student ID of the student you want deleted: 15
Deleting values done

 Students Table:
 1. Add Values
 2. Delete Values
 3. Display Table
 4. Make your own Select Query for students table
 5. Read CSV File to populate students table
 6. Back to Main Menu

Enter corresponding number task to be done: 3
Contents of the table:

Student Table (ID (PK), FirstName, LastName, Email)
(6, 'Abir', 'Das', 'adas@ccny.cuny.edu')
(11, 'Henry', 'Riaz', 'HR@gmail.com')
(12, 'Derrick', 'Duller', 'dd@gmail.com')
(13, 'Johny', 'Mark', 'JM@gmail.com')
(14, 'Mark', 'Wood', 'MW@gmail.com')
(16, 'James', 'Bond', 'JBond@yahoo.com')
```

In this picture, user inputs 2 to delete data from student table. They delete the student with ID 15 and display table by choosing option 3 to check it was actually deleted.

Student table Select Query



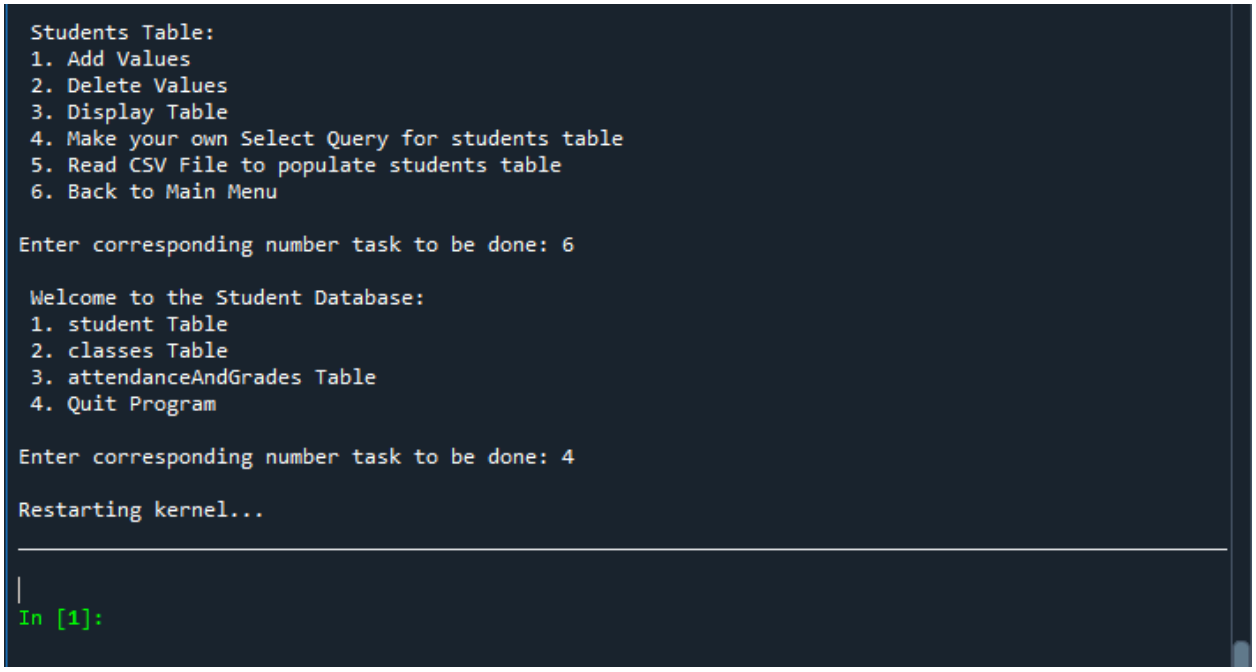
In this picture, user makes their own query (option 5). They want to find last name of a student, given first name is “James” and the program returns “Bond”.

Student table importing csv file

```
 Students Table:
 1. Add Values
 2. Delete Values
 3. Display Table
 4. Make your own Select Query for students table
 5. Read CSV File to populate students table
 6. Back to Main Menu

Enter corresponding number task to be done: 5

Enter file directory of csv file for students table: C:\Users\abir_\Desktop
\studentTablecsvSample.csv
Record inserted
Record inserted
Record inserted

 Students Table:
 1. Add Values
 2. Delete Values
 3. Display Table
 4. Make your own Select Query for students table
 5. Read CSV File to populate students table
 6. Back to Main Menu

Enter corresponding number task to be done: 3
Contents of the table:

Student Table (ID (PK), FirstName, LastName, Email)
(6, 'Abir', 'Das', 'adas@ccny.cuny.edu')
(11, 'Henry', 'Riaz', 'HR@gmail.com')
(12, 'Derrick', 'Duller', 'dd@gmail.com')
(13, 'Johny', 'Mark', 'JM@gmail.com')
(14, 'Mark', 'Wood', 'MW@gmail.com')
(16, 'James', 'Bond', 'JBond@yahoo.com')
(17, 'Abigail', 'Cruz', 'abi@gmail.com')
(18, 'Brandon', 'Smith', 'bran@gmail.com')
(19, 'Ellen', 'Jones', 'ellen@gmail.com')
```

In this image, the user picks option 5, to use a csv file to populate student table. They input the file path and our program inserts the record from the given file. Then the user checks the table to see it actually updated the table or not.

Quitting program



In this picture, the user returns to main menu and then quits the program.